Atmospheric Remote Sensing Data Processing Through Grid Computing Technique

L. Mossucca[1], O. Terzo[1], M. Cucca[2], and R. Notarpietro[2]
[1]Istituto Superiore Mario Boella, Via P. C. Boggio 61, Torino, Italy
[2]Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, Italy

Abstract— *The space-based GPS limb sounding, conventionally known as GPS Radio Occultation, is a remote sensing technique aiming at characterize the Earth atmosphere. The atmospheric profiling is done through the inversion of excess-phase measurements carried-out by a GPS receiver placed on-board a LEO (Low Earth Orbit) satellite. The GPS signal, observed when it emerges from the Earth limb, is refracted by the Earth atmosphere; through inversion techniques, it is possible to retrieve atmospheric refractivity profiles, which in turn, may be used to obtain temperature, pressure and humidity profiles. Considering that there is a great deal of data to process, this paper presents an architecture solution based on Grid Computing. We want to focus the attention on two aspects: how we managed the parallelization of the processing chain of Radio Occultation data and advantages to use this kind of architecture.*

Keywords: Grid Computing, Radio Occultation, GPS, Scheduler, Agent, E-science application

1. Introduction

In the years 2000-2001, the Italian Space Agency (ASI [12]) has developed a new GPS receiver devoted to Radio Occultation, namely ROSA (Radio Occultation Sounder of the Atmosphere). ROSA will be soon operational thanks to a Memorandum of Understanding signed between ASI and the Indian Space Research Organization (ISRO [13]), in which it is agreed that the OCEANSAT-2 mission will carry the Italian GPS Radio Occultation sounder. GPS - Radio Occultation is a quite recent atmospheric remote sensing technique. It is based on the inversion of the Doppler (and, on some extent, of the signal amplitude) time evolution measurable on the GPS signal received on-board a Low Earth Orbit satellite, when the transmitter is setting or rising at the limb horizon. Result of this inversion is the atmospheric refractivity profile which, in turn, can be used to characterize temperature, pressure and humidity profiles. In the framework of this mission, the data derived from occultation events observed by the ROSA instrument will be downloaded by both the Indian and the Italian receiving stations, where it will be processed by the ROSA Data Processing Centre, completely developed by Italian universities, research centres and industrial partners. In particular, this part of the ground segment will be implemented during the first phase by an integrated computing infrastructure installed in Matera and mirrored at Hyderabad in India and, during the second phase, on a distributed software and hardware infrastructure. This second infrastructure will perform the rapid and precise Orbit Determination and Prediction, the unambiguous bending and impact parameters profiles extraction, the Ionospheric correction and the stratospheric initialization, the refractivity, pressure, temperature and humidity profile retrieval, the value added services for meteorology, climate and space weather applications. The data processing will be distributed through the various research centres and universities involved, connected by a GRID computing infrastructure. These nodes are geographically located in several remote places: "Istituto Superiore Mario Boella" (Torino), "Politecnico di Torino" (Torino), "Università di Padova" (Padova), Università "La Sapienza" (Roma), "Università di Camerino" (Camerino), "International Center of Theoretical Physics" (Trieste), "Innova Consorzio per l'Informatica e la Telematica" (Matera), "ASI" (Matera). The software Globus ToolKit has been adopted: it is a middleware application that provides a set of software tools to implement the basic services and capabilities required to built a computational Grid.
This paper is organized as follows: Section 2 discusses the processing chain of RO observations, the scientific algorithms used and output data obtained. Section 3 notices the problems we solved, executing the chain by the Grid. Section 4 presents the structure of our grid architecture proposal. Section 5 explains scheduler and agent implementation, through a detailed description of the propose scheduling algorithms and agent functionalities. Section 6 contains the time execution results obtained by the system based on Grid computing. Section 7 draws the conclusions and emphasizes directions for future work.

2. The processing chain of RO observations

The ROSA observed data, once acquired by the receiving ground station, are processed to produce refractivity, temperature and humidity profiles. The Radio Occultation (RO) events data processing consist of seven main steps, named Data Generators (DGs). All DGs are executed in series with a specific order: the Elaboration step n receives the results

of Elaboration step n-1 as input and its output provides the input for Elaboration step n+1, and so on until all the DGs are executed. Figure 1 represents a diagram of the whole processing chain.

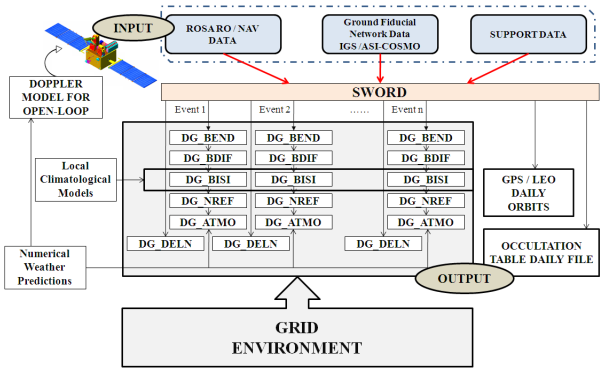

Fig. 1: RO processing chain.

SWOrD (SoftWare for Oceansat-2 oRbit Determination) is at the base of the whole chain process, it acquires ROSA RO Data and ROSA Navigation Data from IGS/GPS ASI with GPS constellation and Oceansat-2, and it works on two levels: first level allows evaluating LEO Rapid Orbits estimated through dynamical methods and GPS Ultra Rapid Orbits and second level allows evaluating the excess phases on both carrier signals L1 and L2 for each occultation event in the input files (ROSA RO and GPS/LEO Orbit). The input and output files cover a 24 hours time interval each, for each day we have about 256 files event to process in Grid in sequential way. It also produces a table which describes the localization of all the occultation events.

The DG_BEND provides raw bending and impact parameter profiles computed on GPS occulted signals on both L1 and L2 GPS frequencies using a Geometrical Optics approach. It processes L1 and L2 Excess Phase data in order to compute bending and impact parameter profiles referred to the atmospheric center of refraction. For each event, this center of refraction is computed on the basis of latitude and longitude of the geometrical tangent point. Bending and impact parameter profiles given as output are of two types: the former is obtained filtering the L1 excess-phase data at a high cut-off frequency; the latter is computed filtering the L1 and L2 excess phase data at a lower cut-off frequency. For each value of the profile, the location of the tangent point (latitude, longitude) and the time tag relative to the time elapsed from the beginning of the occultation are provided.

For each event, the DG_BDIF provides a bending and impact parameter profile on which the ionospheric effects have been compensated. It processes both L1 and L2 Bending and impact parameter profiles derived from DG_BEND, in order to minimize the first order ionospheric dispersive effects.

The DG_BISI provides bending angle and impact parameter profiles optimized with climatological values of atmospheric parameters in stratosphere, where bending angle and impact parameter measured values are more fluctuating. It processes the bending angle and the Impact Parameter profiles obtained from DG_BDIF in order to smooth the data in the higher layers of the profile. This smoothing process is obtained calculating the value of the bending angles from climatological values of pressure and temperature for the geographic coordinates and for the local time of the occultation.

For each event, the DG_NREF provides the refractivity profile and the dry air temperature and pressure profiles. It is able to invert iono-free and properly initialized bending and impact parameter profiles in order to compute the correspondent "quasi" vertical refractivity profile. In addition, dry air temperature and pressure profiles shall be available after this processing stage. The location of each point of the profile will give geodetic latitude and longitude in a global domain.

The DG_ATMO allows retrieving wet temperature and water vapour pressure profiles. The evaluation of water vapour is performed using a background profile obtained from Global Climatological Models or Numerical Weather Prediction.

The DG_DELN allows evaluating the electronic density profile in ionosphere. It produces an output file containing the electronic density vertical profile related to the observed occultation. The location of the tangent point (latitude, longitude and altitude) and the time tag are given for each value of the profile.

3. Context consideration

We estimated that every day the number of events is approximatively 256 and each of them must process executing all DGs in series. In fact in this context the first difficulty is to find a solution that reduces the total time processing for all events because in a non distributed context every event must be executed in series. The second characteristic that we must solve concerns the availability and the capacity to process all events. In a non distributed context the entire process depend on the availability of the node in charge to execute every DG and it could create a bottleneck problem. The main idea was to create a flexible architecture in condition to process the entire processing chain for all events. In consideration of this we think that a distributed architecture is the best choice for a couple of reasons. First of all is the entire chain process could be executed on one node or in different nodes. This feature guarantees that if a piece of chain has been executed in a node and for some reasons this node became unavailable the schedule policies could switch the job processing on a different node. In this context the goal is to obtain a system that process several events in parallel mode, each following a specific chain process execution. Another reason is economic, because each partner needn't a license for every software used by DGs so this solution allows to every user to obtain result executing DGs in a remote machine. Finally in term of scalability in our specific case the goal is to develop

two agent components resident on each node sending some information to grid master node used by the scheduler to choose the machine which will execution DG.

4. System components proposal

4.1 Overview

The Grid Processing Management (GPM) is an integrated system devoted to handle and process RO data of the OCEANSAT-2 ROSA on board sensor. The GPM is

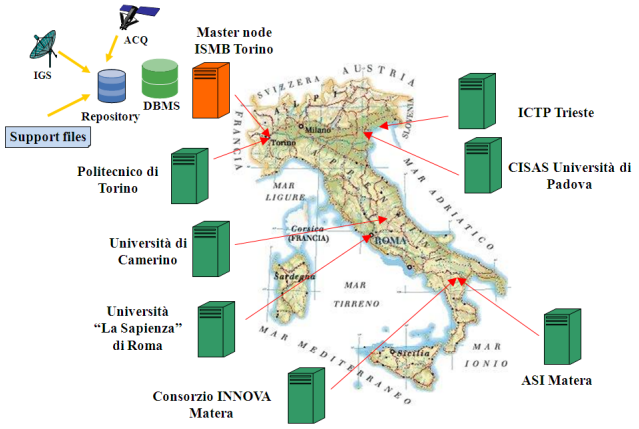

Fig. 2: Architecture overview.

composed by the following subsystems (Figure 2):

- Middleware;
- GARR Network;
- Central Repository;
- Relational Database;
- Adapter;
- Local Scheduler;
- Agents;
- Application.

First of all, we want to explain that the general purpose of our project is: to share the computational resources, to transfer a great deal of files and to submit jobs from several different organizations of the scientific community located in different places in Italy. Hence, we decided to use the Globus Toolkit as middleware [1] [10] [11], since it allows obtaining a reliable information technology infrastructure that enables the integrated, collaborative use of computers, networks and databases.

Another instrumental component on the Grid is certainly the network. In our project since it involves several research Italian centers, we had the opportunity to use the GARR. The GARR (Gruppo per l'Armonizzazione delle Reti della Ricerca [14]) is the Italian network, born in 1977 for the research. It is designed to be scalable, fast and reliable and it allows data link on demand in 10 Gbit/s with in the GARR community.

The Central Repository is the component responsible to store data with directories and files structure after processing.

The Relational Database is deputed to store processing chain configuration, information about the location of raw data and products, software needed for execution of each DG, grid configuration and information to write XML files. It stores also information about users account, information about machines belonging to the Grid and elapsed time for each execution.

The Adapter reads the XML files and prepares the working directories and input files for data generators execution by providing a copy on the node selected for the process execution from the Central Repository. Once the DG has finished its own execution, the Adapter stores the output file on the Central Repository and update the Relational Database.

4.2 Java application

In this preliminary step of development, where our algorithms are continuously improved, it was necessary an user friendly Java Application (Figure 3). It was born for researchers of university in order to allow them to test its own algorithms but it also provides some functionalities:

- to grant secure access to the Grid;
- to allow choosing where DG will be executed, in local machine or in Grid through the scheduler;
- to display the evolution of the submitted DG;
- to allow monitoring available machine belonging to the Grid;
- to plot the final result of DG.

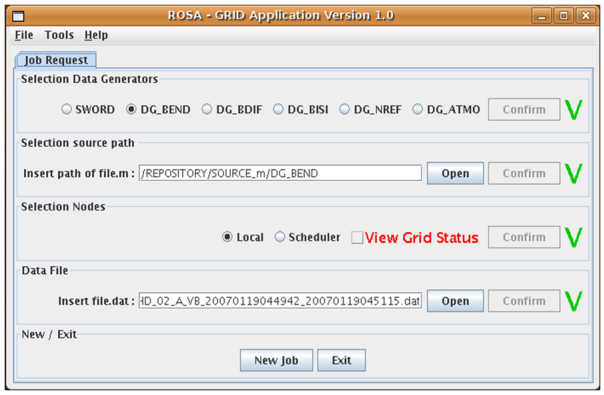

Fig. 3: Job submission application.

At a later time a procedure will be available to process chain in automatic mode, it will be able to generate POD data, RO level 2 and 3 and advanced products for space geodesy as soon as row data is downloaded from satellites.

5. Job scheduling model

As explained before the chain process is composed of 6 jobs, from the orbital prediction and determination to

the atmospherics parameters. The entire chain processing involves complex algorithms which use a set of languages as Fortran, MatLab, C++, Mathematica, Java and Perl and library as libxerces. In this context we decided to develop an ad-hoc job management scheduler because after several tests with centralized schedulers like PBS, Condor and decentralized ones like GridWay, we considered, because of the complexity of scenario, that they didn't satisfy our requirements. For this reason we developed two system agents and a specific scheduler.

5.1 Agent

In spite of the success of Grid computing in providing solutions for a number of large scale science fields one of the problems is the scalability of the system. The agents are emerging as a solution to provide flexibility and scalability [4]. In fact the first requirement for the scheduler is that it never requests information on each nodes to take a decision to schedule the jobs. In our architecture are present two types of agents: Job and System agent.

The first one is used to monitor the behavior of some parameters, as CPU,RAM and swap on nodes during the DG execution, it has been started from Java application and when DG is finished, it handles sending these information to master node.

The second one, the system agent, installed on each node, is used to monitor the availability of each service on the node and periodically, every 5 minutes, it sends to the database on the grid master node its general status, if all services are available and if the node is in condition to receive a job. In this case the advantage is that the scheduler only queries the database for the pre selection of the nodes list in condition to receive a job. A specific function checks if the feedback information from all nodes has been sent and, in case of missing status, the node is considered not available.

5.2 Scheduling model

We split the scheduler functions in three phases [8], after it received a job request:

The resources' discovery, the phase is directly related to the agents information sent by the agents and the node discovery is executed in a short time. This first step return a pre list of nodes.

The second phase, name information gathering, depends directly on the job type requested and there is a check of the softwares and applications needed by the job and the software capabilities of each node preselected. we define this phase the broker function, it handles balance what we need with what we have.

In the third phase, we look at the pre selected node list in order to verify which nodes are free for a new execution and to assign the job.

In Figure 4 the comparison of processing time between the two different architectures is depicted: in both examples the

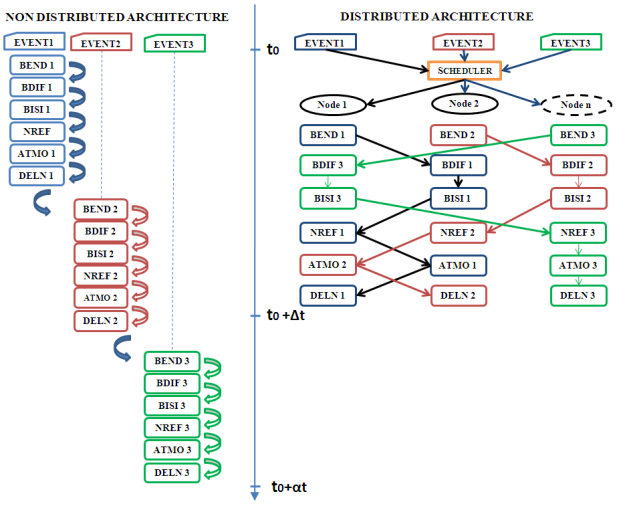

Fig. 4: Processing-time comparisons.

$$t_0 + \Delta_t = EndProcessingTime(DA)$$

$$t_0 + \alpha t = EndProcessingTime(NDA)$$

start time is set to t_0, but while in the DA event execution terminates at $t_0 + \Delta_t$, in NDA it finishes at $t_0 + \alpha_t$, with $\Delta_t < \alpha_t$. This time reduction is attributable to the fact that in the DA it is not necessary to execute all the jobs related to one event in the same node. Instead, it is possible to run, for example, in Node 1 the BEND job from Event1 but also in the same time the BEND job from Event2 on Node 2.

5.3 Queueing approach

In phase of analysis specific attention was dedicated to the queue approach and how to manage the queue of jobs request. In term of length of execution time for each DGs we have implemented a learning phase from all DGs over all nodes. In this way we had a clear indication of the process duration time for a specific DG on each node. This is an important feature on our scenario because the panel of nodes cover Dual Core and Quad Core capabilities. In fact, considering the large amount of events, estimated 256 by day, and for each event the characteristic of the chain process, it is not so unfrequent that we had all nodes in running condition, in this case we minimize the end time of jobs running on each machine, we look the start time, the duration estimated for the job and we select the node that finishes first his running job by inserting the new job requested in queue.

6. Experiment results

After some test, where we considered effective the number of files in input and in output, metadata, size and elaboration time, we wanted present the difference time of execution

using a non distributed and distributed architecture. In an architecture with only one machine, the time of execution is:

$$T_p = Ts + (Te * \eta) \quad (1)$$

In an architecture with more machines, the time of execution is:

$$T_p = (Ts + \beta_s) + \frac{(Te + \beta) * \eta}{N} \quad (2)$$

Where:

$$T_p = TotalTimeProcess \; T_s = SwordTimeProcess$$
$$T_e = EventTimeProcess \; \eta = ROEventNumber$$
$$N = GridNodeNumber \; \beta = TotalFileTransfertime$$
$$\beta_s = SwordFileTransfertime$$

In Table 1, we can see estimation time for execution of a single event in a non distributed architecture, in this case there is any transfer time.

DG	TIME (mm)
BEND	4.30
BDIF	2.30
BISI	1.00
NREF	0.50
ATMO	6.00
	14.50

Table 1: Estimation time for a single event (NDA)

In Table 2 instead, we can see estimation time for execution of a single event in a distributed architecture, in this case there is transfer time.

DG	TIME (mm)
BEND	5.23
BDIF	2.45
BISI	1.31
NREF	2.18
ATMO	7.36
	19.33

Table 2: Estimation time for a single event (DA)

In Figure 5, comparison between execution in distributed and non distributed architecture is depicted, as we can notice, in the time execution decreases when we increase the number of nodes. We observed that, in a non distributed architecture, the total execution time is 3712 minutes (about 60 hours), instead in a distributed, as of about 41 hours, if two nodes are available, increasing the number of nodes, the execution decrease further. We want to point out that we don't get any gain time in grid environment when a single event is processed. Actually it spends more time because we need to add transfer files time. In fact we have a sizeable gain time only when we process a collection of files.

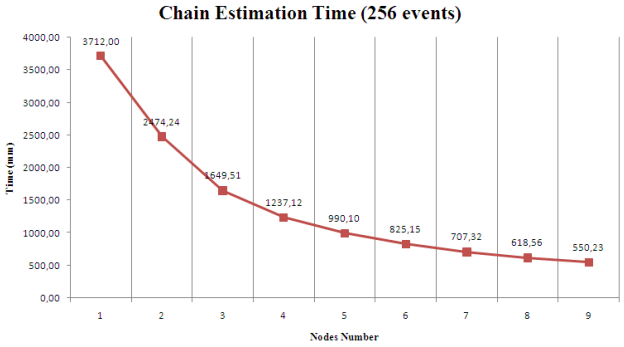

Fig. 5: Comparison of execution.

7. Conclusions and future works

In this paper we have presented a new scenario for Radio Occultation data processing based on Grid Computing. First, we have explained which problem could present, then we have introduced the components of our system. We have focused on job scheduling which involves interactions between scheduler and agent. We have also discussed the possibility to share results on scientific community. In future, we think it will be possible create DGs compiled in order to be executed on each node without considering library and software installed. This solution decreases the cost of software license. Instead, to increase the computing power we are considering that the possibility to create a cluster behind each node, so to make a hierarchical structure of Grid.

8. Acknowledgments

The authors are grateful to Italian Space Agency (ASI) for supporting this project within contract I/006/07/0 and to all the ROSA-ROSSA partners for their contributions.

References

[1] Berman, Fox, Hey, *Grid Computing Making the Global Infrastructure a Reality*, Wiley, Ed. , 2005.
[2] Foster, Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publisher Inc , 2003.
[3] Jacob, Ferreira, *Enabling Application for Grid Computing with Globus*, IBM RedBooks, Ed. , 2003.
[4] Kurowski, Nabrzyzki *Scheduling Jobs on the Grid- Multicriteria Approach*, Computational Methods in Science and Technology , 2006.
[5] Hassaine, *Issues in Selecting a Job Management System*, Sun Microsystems, Ed. , 2002.
[6] Gradwell *Grid Scheduling with Agents*, Department of Computer Science, University of Bath , 2003.
[7] Hongzhang, Warren, Oliker, Biswas *Job Scheduling in a Heterogeneous Grid Environment*, , 2004.
[8] Buyya, R. Abramson, D. Giddy *An architeture of a resource management and scheduling system in a global computational Grid*, , 2000.
[9] Java CoGKit. [Online]. Available: http://wiki.cogkit.org/index.php/Main Page
[10] Globus. [Online]. Available: http://www.globus.org/
[11] Globus. [Online]. Available: http://www.globusconsortium.org/
[12] Agenzia Spaziale Italiana, ASI. [Online]. Available: http://www.asi.it/

[13] Indian Space Research Organization, ISRO. [Online]. Available: http://www.isro.org/
[14] Gruppo per l'Armonizzazione delle Reti della Ricerca, GARR. [Online]. Available: http://www.garr.it/